

Window 환경에서의 CI/CD 실습

실습목적: window 환경에서 Jenkins 및 Tomcat 9 설치하고 github에 push (지속적인 통합:CI)될 때 자동으로 Tomcat 9에 SpringBoot 어플리케이션을 war로 배포(지속적인 배포:CD)하는 실습이다.

1. JDK 11 설치

1) 환경변수 설정

JAVA_HOME= C:\Program Files\Java\jdk-11

PATH=%JAVA_HOME%\bin;%PATH%

2) 환경변수 설정 후 확인하기

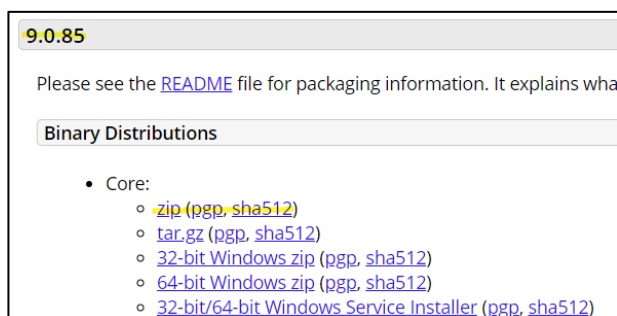
```
C:\Users\inky4>set JAVA_HOME
JAVA_HOME=C:\Program Files\Java\jdk-11

C:\Users\inky4>set PATH
Path=C:\Program Files\Java\jdk-11\bin;C:\Program Files\Java\javapath;C:\Program Files\nodejs\;C:\
```

2. Tomcat9 설치

1) 다운로드

<https://tomcat.apache.org/download-90.cgi>



(1) conf/server/xml 에서 port 번호 변경

```
69     <Connector port="8090" protocol="HTTP/1.1"
70               connectionTimeout="20000"
71               redirectPort="8443"
72               maxParameterCount="1000"
73     />
```

<Valve> 태그 주석 처리

```
<Context antiResourceLocking="false" privileged="true" >
  <CookieProcessor className="org.apache.tomcat.util.http.Rfc6265CookieProcessor"
    sameSiteCookies="strict" />
  <!-- Valve className="org.apache.catalina.valves.RemoteAddrValve"
    allow="127\.\d+\.\d+\.\d+:::1|0:0:0:0:0:0:0:1" / -->
  <Manager
    sessionAttributeValueClassNameFilter="java\.lang\.(?:Boolean|Integer|Long|Number
    \.Literal|\.filters\.CsrfPreventionFilter\$LruCache(?:\$1)?|java\.util\.(?:Linked
  </Context>
```

<Valve> 태그 주석 처리

```
<Context antiResourceLocking="false" privileged="true" >
  <CookieProcessor className="org.apache.tomcat.util.http.Rfc6265CookieProcessor"
    sameSiteCookies="strict" />
  <!--Valve className="org.apache.catalina.valves.RemoteAddrValve"
    allow="127\.\d+\.\d+\.\d+|::1|0:0:0:0:0:0:1" / -->
  <Manager
    sessionAttributeValueClassNameFilter="java\.lang\.(?:Boolean|Integer|Long|Number
    \.catalina\.filters\.CsrfPreventionFilter\$LruCache(?:\$1)?|java\.util\.(?Linked
  </Context>
```

(4) conf/tomcat-users.xml 파일에 사용자 권한 및 계정 추가

```
<role rolename="manager-gui"/>
<role rolename="manager-script"/>
<role rolename="manager-jmx"/>
<role rolename="manager-status"/>
<user username="admin" password="admin" roles="manager-gui,manager-script,manager-jmx,manager-status"/>
<user username="deployer" password="deployer" roles="manager-script"/>
<user username="tomcat" password="tomcat" roles="manager-gui"/>
```

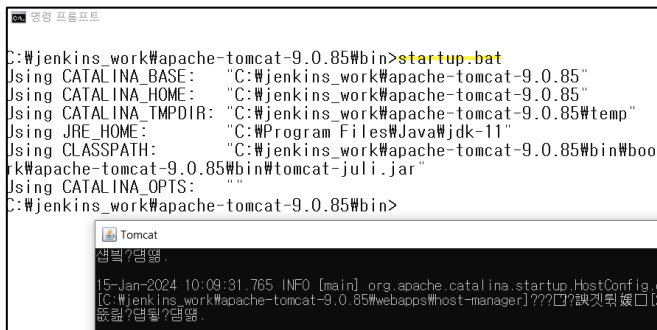
```
<role rolename="manager-gui"/>
<role rolename="manager-script"/>
<role rolename="manager-jmx"/>
<role rolename="manager-status"/>
<user username="admin" password="admin" roles="manager-gui,manager-script,manager-jmx,manager-status"/>
<user username="deployer" password="deployer" roles="manager-script"/>
<user username="tomcat" password="tomcat" roles="manager-gui"/>

</tomcat-users>
```

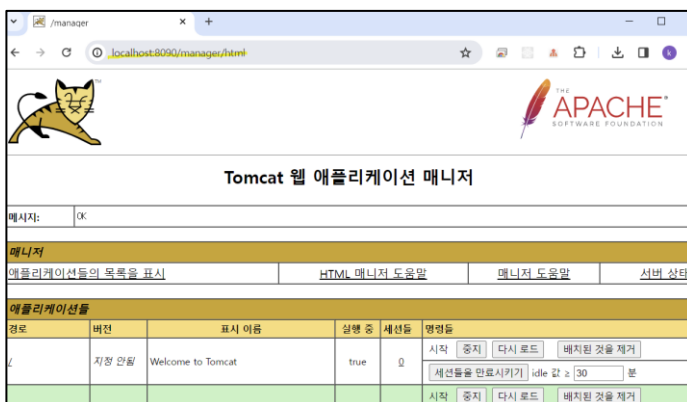
(5) tomcat 서버 실행 및 종료

서버시작: bin/startup.bat

서버종료: bin/shutdown.bat



<http://localhost:8090/manager> 요청후 admin/admin 입력.

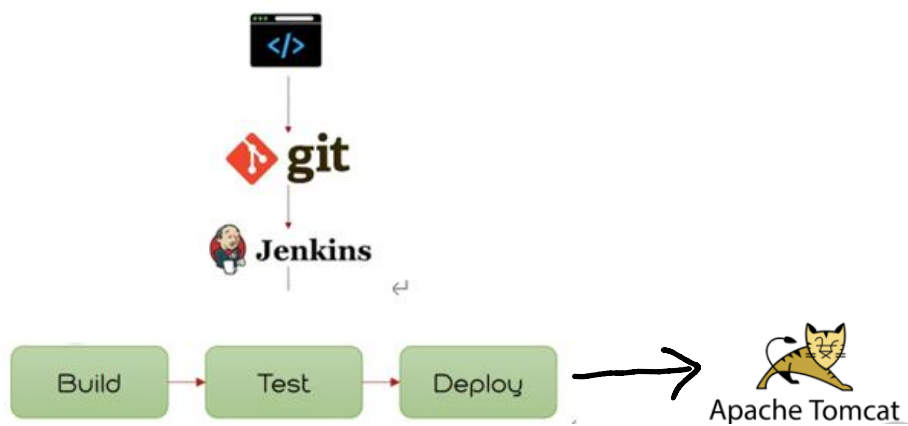


3. Jenkins 설치

1) 특징

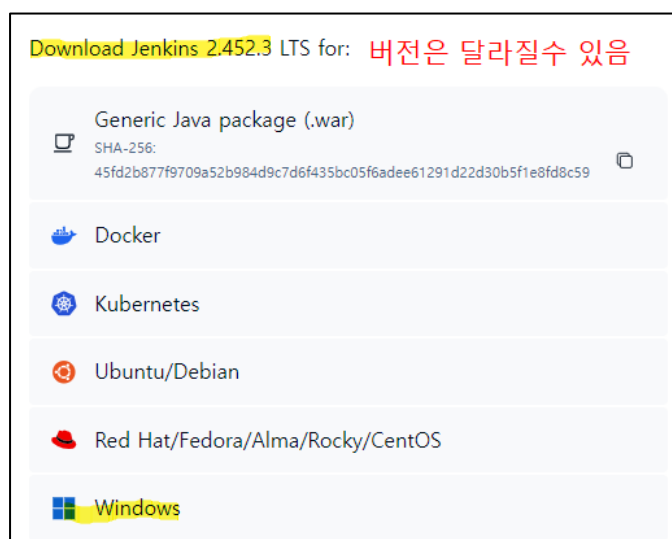
- 지속적인 통합과 배포
- 다양한 plugins 연동 (Git, Maven, Java 등)

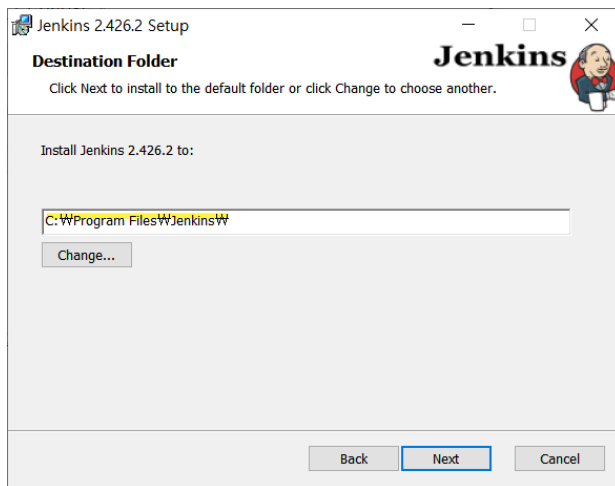
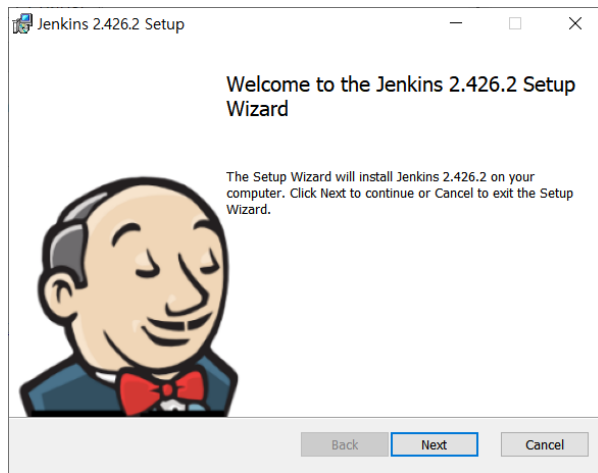
2) 아키텍처



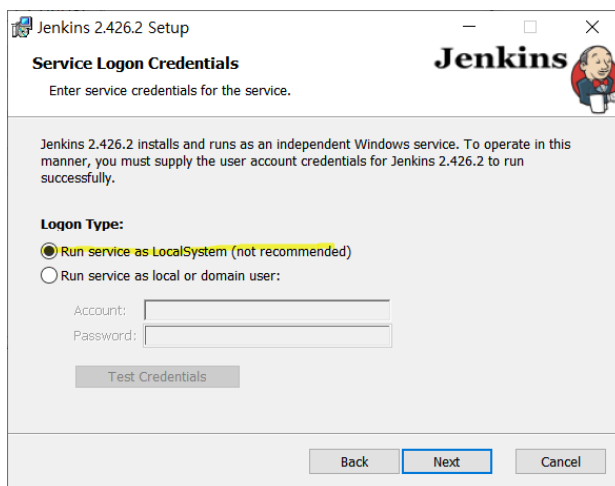
3) Jenkins 설치 (window 버전)

<https://www.jenkins.io/download/>

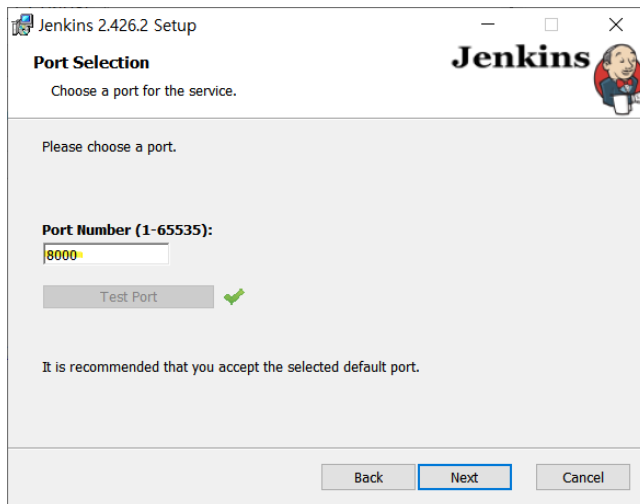




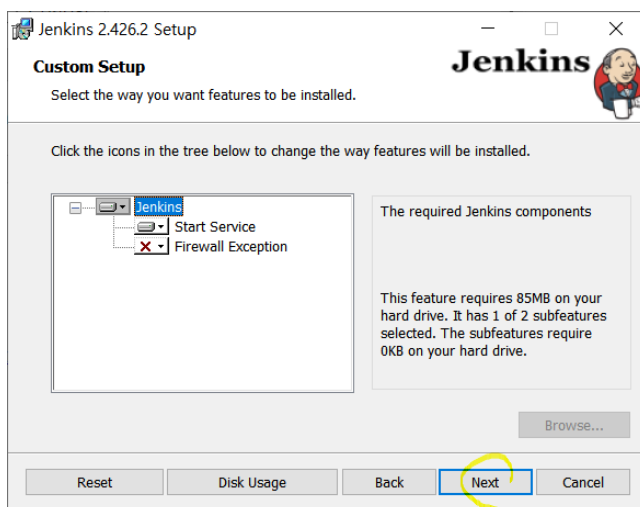
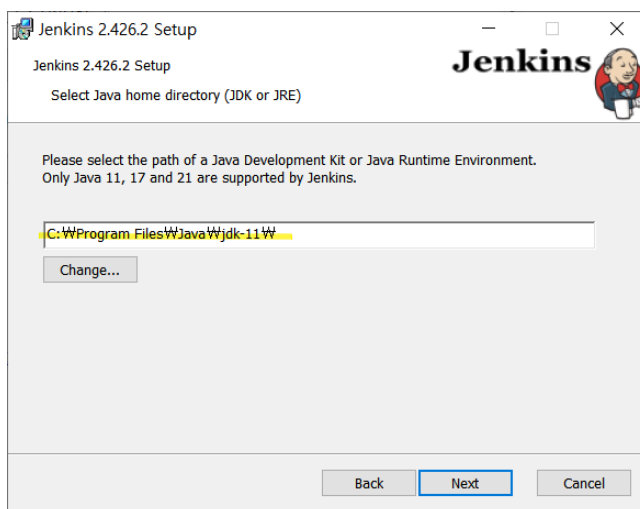
Logon Type 은 Run service as LocalSystem 을 선택.

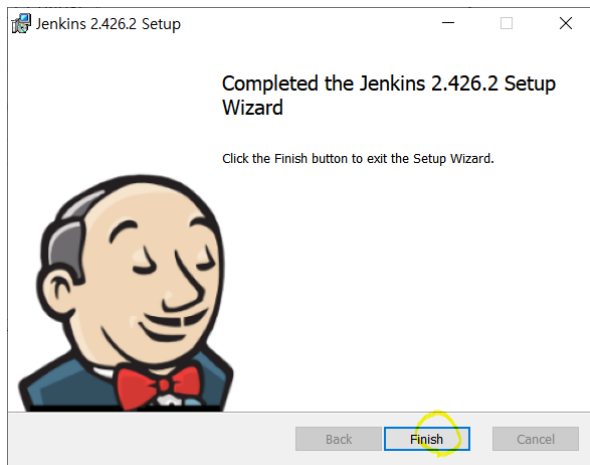
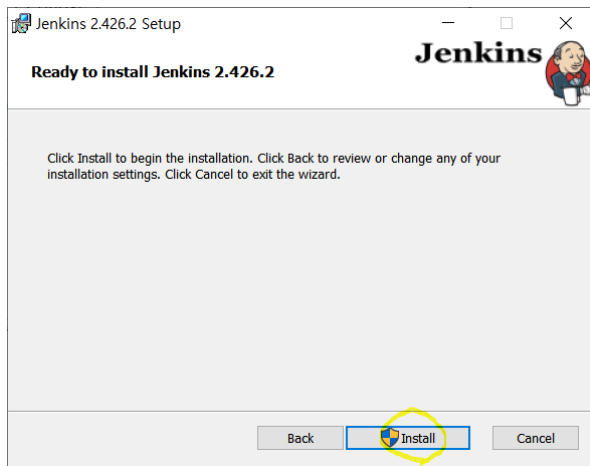


Jenkins의 기본 port는 8000 임.



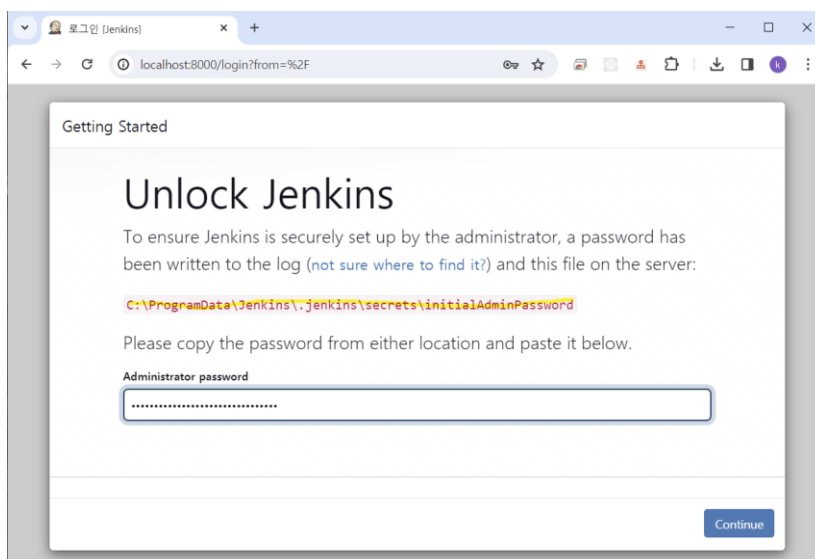
설치된 JDK 경로 지정됨.





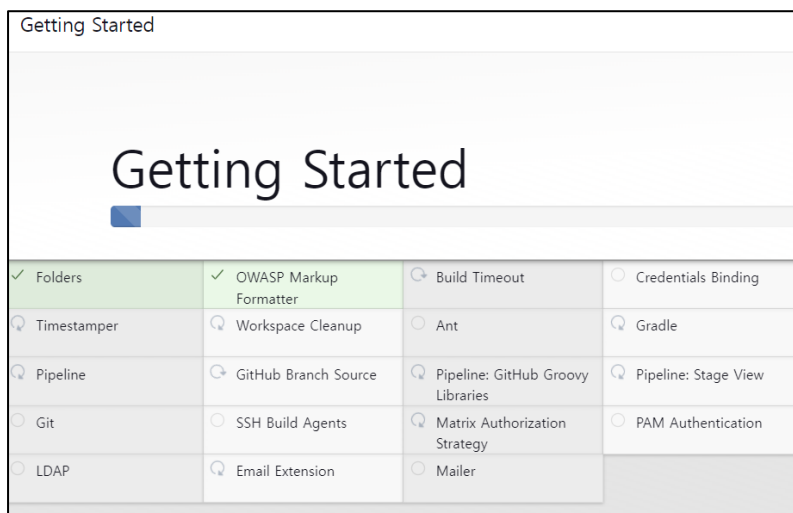
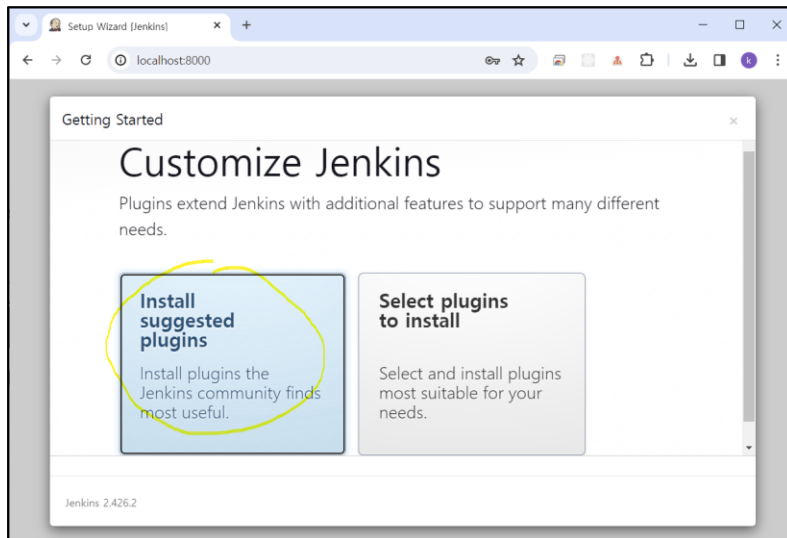
4) Jenkins 서버 요청

<http://localhost:8000> 요청하면 다음과 같은 비번입력 초기화면이 출력된다.



C:\ProgramData\Jenkins\jenkins\secrets\initialAdminPassword 경로에서 비번 복사해서 사용.

5) 플러그인 설치



6) 계정 생성

계정명: admin

암호: admin

Getting Started

Create First Admin User

계정명

admin

암호

..... admin

암호 확인

.....

이름

Administrator

이메일 주소

inky4832@daum.net

Jenkins 2.452.3

Skip and continue as admin

Save and Continue

Getting Started

Instance Configuration

Jenkins URL:

http://localhost:8000/ http://localhost:8000/

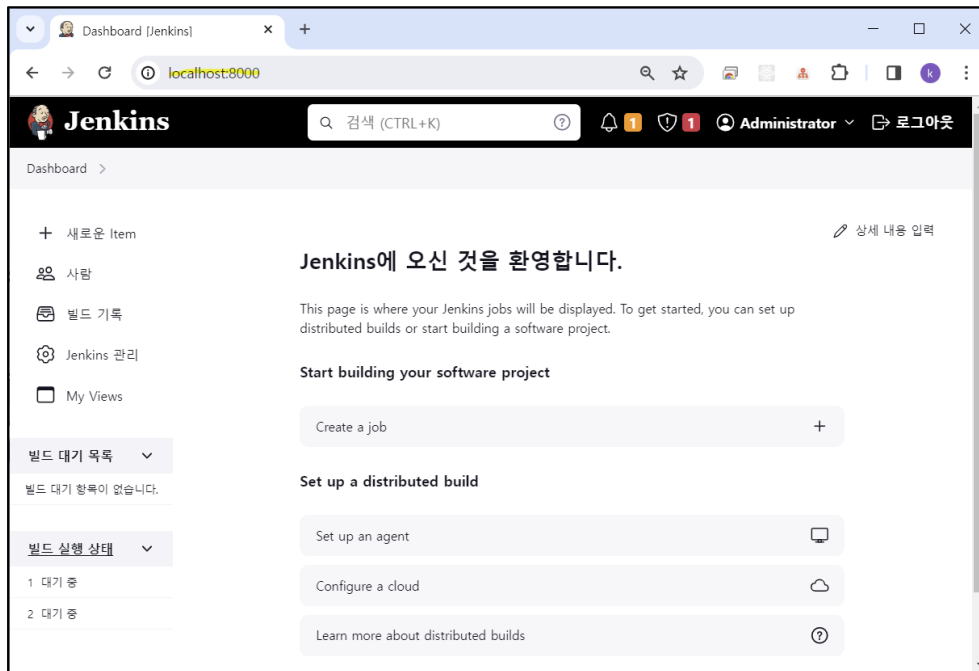
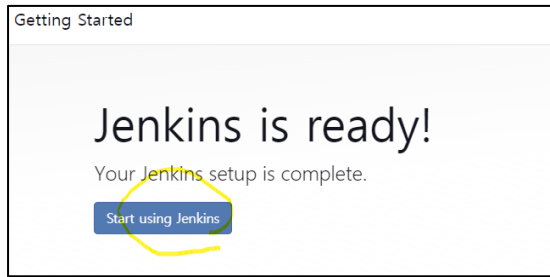
The Jenkins URL is used to provide the root URL for absolute links to various Jenkins resources. That means this value is required for proper operation of many Jenkins features including email notifications, PR status updates, and the `BUILD_URL` environment variable provided to build steps.

The proposed default value shown is **not saved yet** and is generated from the current request, if possible. The best practice is to set this value to the URL that users are expected to use. This will avoid confusion when sharing or viewing links.

Jenkins 2.452.3

Not now

Save and Finish



설치한 Jenkins 를 uninstall 할 때는 uninstall 후에 반드시 다음 2개의 디렉터리 폴더까지 명시적으로 삭제한다.

C:\Program Files\jenkins 폴더 삭제

C:\ProgramData\jenkins 폴더 삭제

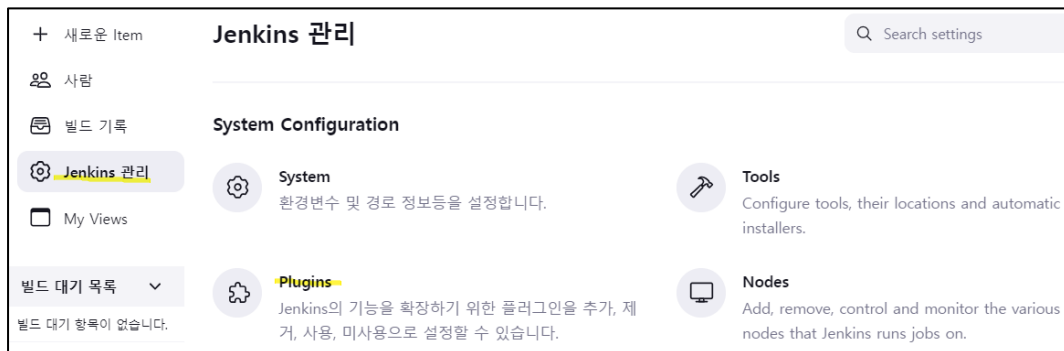
4. Jenkins에 Git과 Maven 설정하기

Github에 저장된 SpringBoot Todo 어플리케이션을 참조해서 Jenkins에 war 저장.

(명시적으로 Jenkins 에서 [지금 빌드 항목]을 선택해야 된다.)

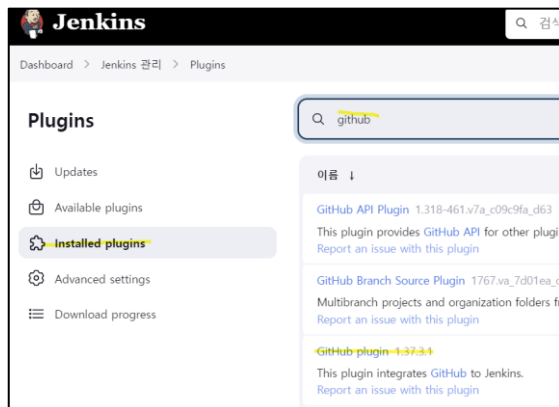
1) git plugin 추가 (Jenkins 관리 > Plugins)

과거에는 git 플러그인을 명시적으로 추가해야 했으나 현재는 자동으로 플러그인이 추가되어있음

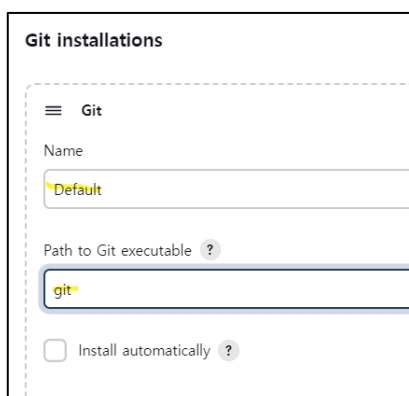


다음과 같이 github 플러그인이 미리 설치되어 있음을 확인할 수 있다.

Plugins > Installed plugins > github 로 검색



플러그인이 설치되었기 때문에 Git 을 설치할 수 있다. 현재는 설치되어 있음 (Jenkins 관리 > Tools)

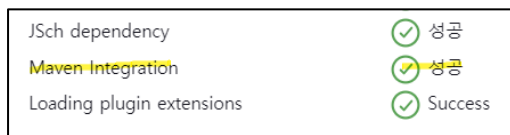


2) Maven Plugin 추가 (maven은 기본으로 설치되어 있지 않음, Jenkins 관리 > Plugins)

Available plugins > maven 검색하고 Maven Integration 선택하고 install 클릭.



설치가 성공하면 하단에 다음과 같이 Success 출력됨.

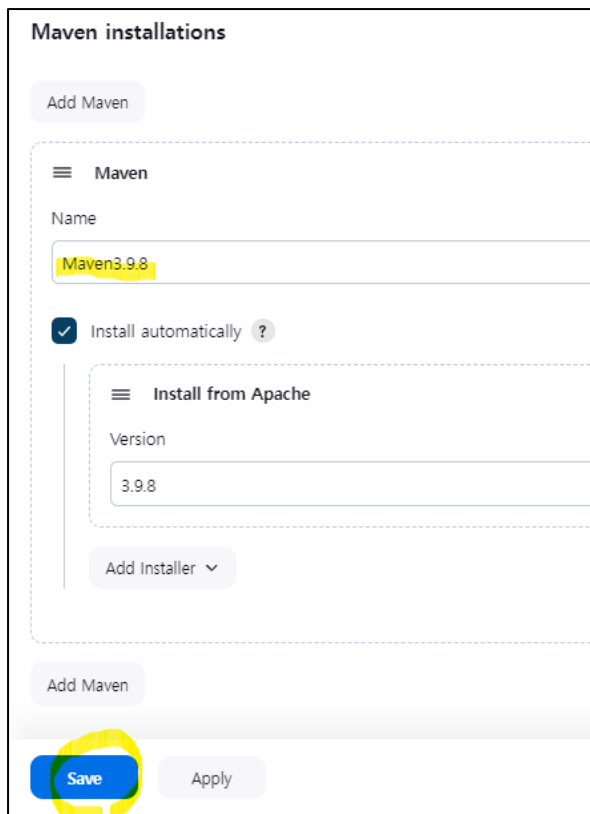


플러그인이 설치되었기 때문에 이제부터 maven 을 설치할 수 있다. (Jenkins 관리 > Tools)

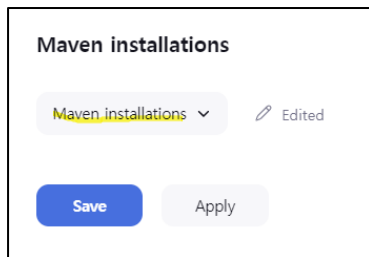
다음 화면에서 Add Maven 버튼을 선택.



Name: Maven3.9.8 입력하고 Save 버튼을 선택하면 설치됨.

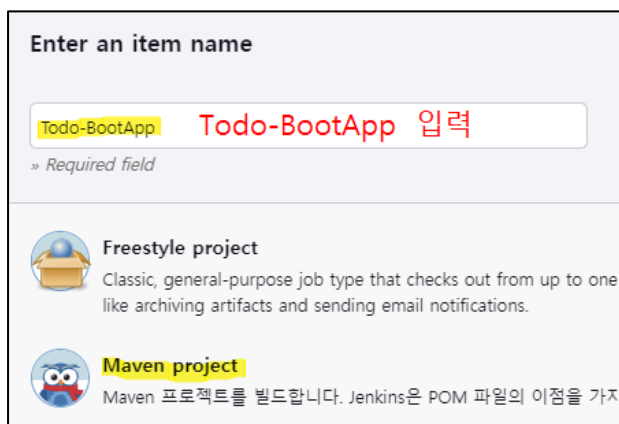


다음과 같이 Maven이 설치된 것을 확인할 수 있다.



3) Jenkins 실행단위인 Item 추가하기 (+ 새로운 Item > Todo-BootApplication 입력)

Maven을 설치했기 때문에 Maven Project를 선택한다.



4) Git 저장소 지정 및 Build 설정

- SpringBoot 빌드시 web.xml 에 해당하는 코드 작성 필요.

@SpringBootApplication

```
public class Application extends SpringBootServletInitializer{  
    public static void main(String[] args) {  
        SpringApplication.run(Application.class, args);  
    }  
    @Override  
    protected SpringApplicationBuilder configure(SpringApplicationBuilder builder) {  
        return builder.sources(Application.class);  
    }  
}
```

- pom.xml 에서 패키징 정보와 파일명 지정 및 mapper 설정 필요

```

<parent>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-parent</artifactId>
  <version>2.7.18</version>
  <relativePath/> <!-- lookup parent from repository -->
</parent>
<groupId>com.exam</groupId>
<artifactId>boot2.7_REST01_basic</artifactId>
<version>0.0.1-SNAPSHOT</version>
<packaging>war</packaging>      war 로 패키징
<name>boot2.7_REST01_basic</name>
<description>Demo project for Spring Boot</description>
<properties>
  <java.version>11</java.version>
</properties>

<build>
  <finalName>todo</finalName>   컨텍스트명으로 활용됨
  <plugins>
    <plugin>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-maven-plugin</artifactId>
    </plugin>
  </plugins>
  <resources>
    <resource>
      <directory>src/main/resources</directory>
      <filtering>true</filtering>
      <includes>
        <include>*/application*.properties</include>
      </includes>
    </resource>
    <resource>
      <directory>src/main/java</directory>
      <includes>
        <include>*/*.properties</include>
        <include>*/*.xml</include>
      </includes>
    </resource>
  </resources>      mybatis mapper 인식
</build>

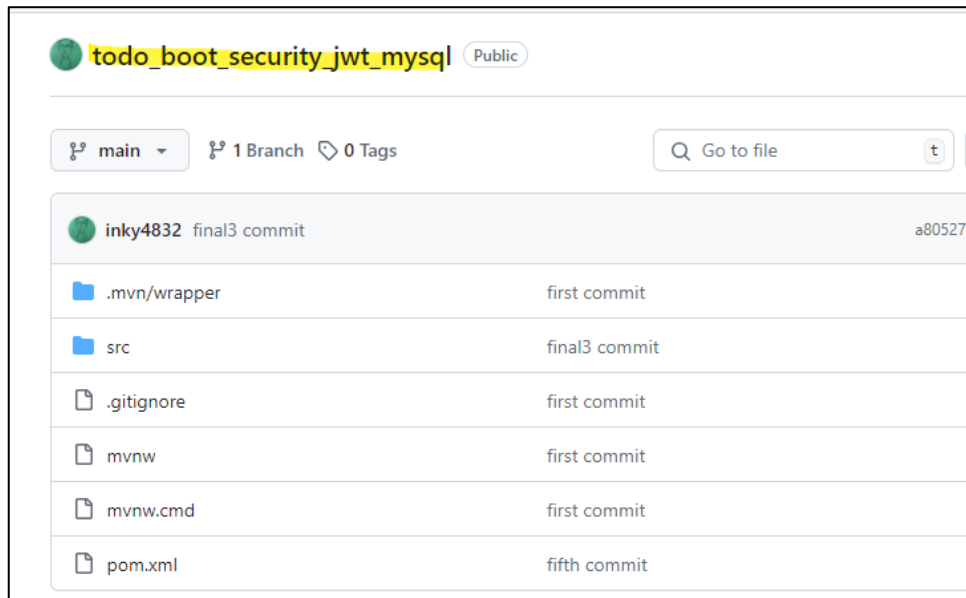
```

위 코드중에서 mybatis mapper 인식 설정은 local 환경에서는 필요 없으나 Jenkins로 war로 패키징 하여 배포시 mapper에 해당하는 xml이 패키징에서 누락되는 현상이 발생됨.

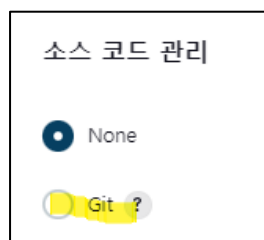
따라서 명시적으로 포함시켜야 된다.

Jenkins 에서 Github에 push된 소스코드를 배포하기 위해서 URL 경로를 지정한다.

https://github.com/inky4832/todo_boot_security_jwt_mysql.git



Todo-BootApp > 구성(Configure) > General > 소스 코드 관리 항목



Git 체크하고 다음과 같이 설정한다.



Branches to build ?

Branch Specifier (blank for 'any') ?

master 값을 main으로 변경

Add Branch

Repository browser ?

Additional Behaviours

Add ▾

스크롤해서 **Build > Goals and Options** 항목에 **clean compile package** 입력

Build

Root POM ?

Goals and options ?

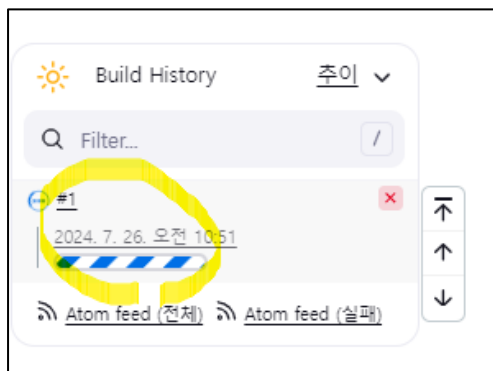
고급 ▾

[저장] 버튼 클릭.

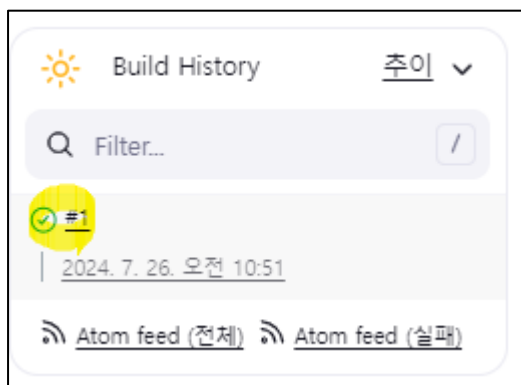
Todo-BootApplication > 지금 빌드 항목 선택하여 빌드한다.



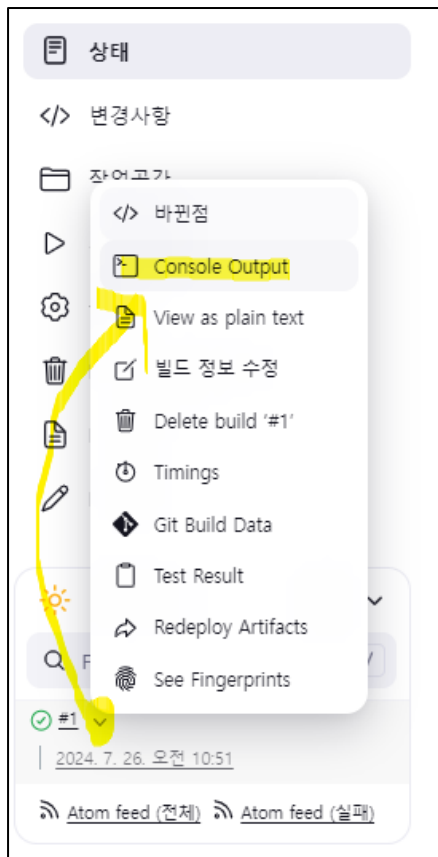
다음과 같이 빌드 중임을 알 수 있음.



이상없이 빌드되면 다음과 같은 화면이 출력됨.



실행결과 로그를 확인하기 위하여 Console Output 선택.

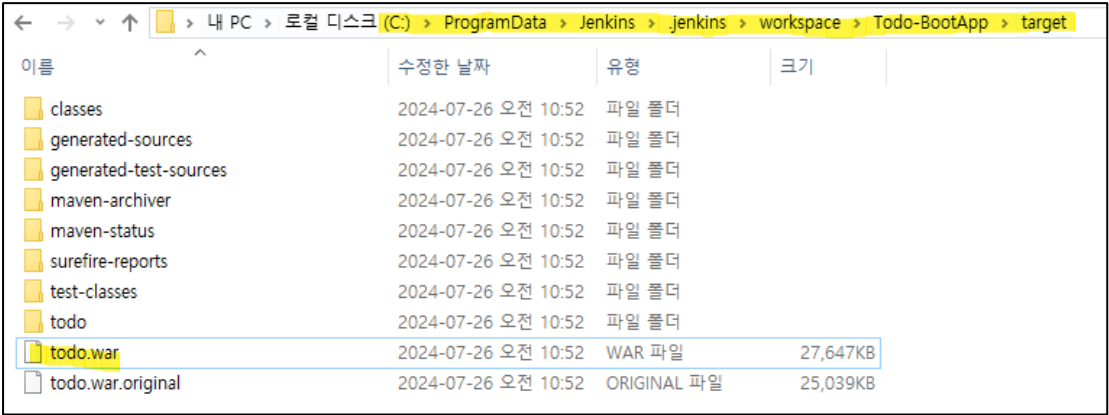


다음과 같이 todo.war 파일이 생성되고 SUCCESS 출력이 된다.

```
[INFO] Processing war project
[INFO] Building war: C:\ProgramData\Jenkins\.jenkins\workspace\Todo-BootApp\target\todo.war
[INFO]
[INFO] --- spring-boot:2.7.18:repackage (repackage) @ boot2.7_REST01_basic ---
[INFO] Replacing main artifact with repackaged archive
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 27.915 s
[INFO] Finished at: 2024-07-26T10:52:23+09:00
[INFO] -----
Waiting for Jenkins to finish collecting data
[JENKINS] Archiving C:\ProgramData\Jenkins\.jenkins\workspace\Todo-BootApp\pom.xml to
com.exam/boot2.7_REST01_basic/0.0.1-SNAPSHOT/boot2.7_REST01_basic-0.0.1-SNAPSHOT.pom
[JENKINS] Archiving C:\ProgramData\Jenkins\.jenkins\workspace\Todo-BootApp\target\todo.war to
com.exam/boot2.7_REST01_basic/0.0.1-SNAPSHOT/boot2.7_REST01_basic-0.0.1-SNAPSHOT.war
channel stopped
Finished: SUCCESS
```

윈도우에서 생성된 todo.war 파일을 조회한다.

C:\ProgramData\Jenkins\jenkins\workspace\Todo-BootApp\target\todo.war



이름	수정된 날짜	유형	크기
classes	2024-07-26 오전 10:52	파일 폴더	
generated-sources	2024-07-26 오전 10:52	파일 폴더	
generated-test-sources	2024-07-26 오전 10:52	파일 폴더	
maven-archiver	2024-07-26 오전 10:52	파일 폴더	
maven-status	2024-07-26 오전 10:52	파일 폴더	
surefire-reports	2024-07-26 오전 10:52	파일 폴더	
test-classes	2024-07-26 오전 10:52	파일 폴더	
todo	2024-07-26 오전 10:52	파일 폴더	
todo.war	2024-07-26 오전 10:52	WAR 파일	27,647KB
todo.war.original	2024-07-26 오전 10:52	ORIGINAL 파일	25,039KB

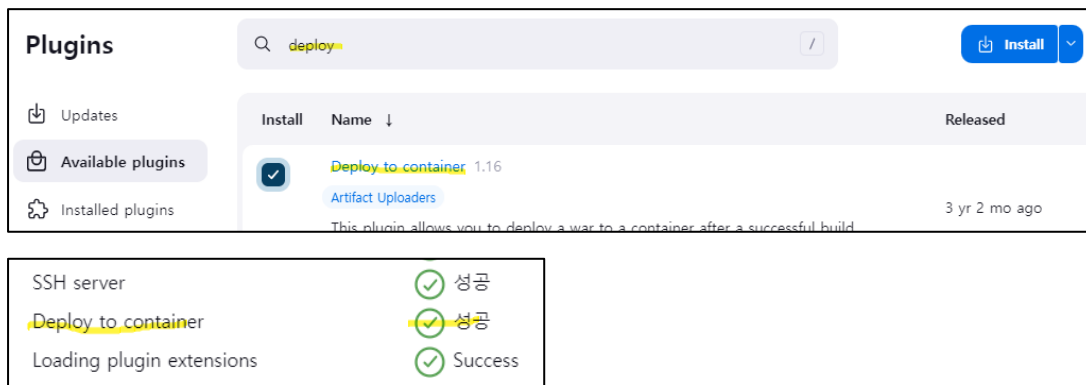
지금까지의 실습 결과는 Github에 push된 Todo 어플리케이션을 Jenkins에서 접근해서 Jenkins가 설치된 PC에 todo.war로 패키징 되었음.

5. Jenkins에 Tomcat9 설정하기

Github에 저장된 웹 어플리케이션을 참조해서 Jenkins에 told.war로 저장하고 사용중인 Tomcat9 서버에 배포까지 처리. (명시적으로 Jenkins 에서 [지금 빌드 항목]을 선택해야 된다.)

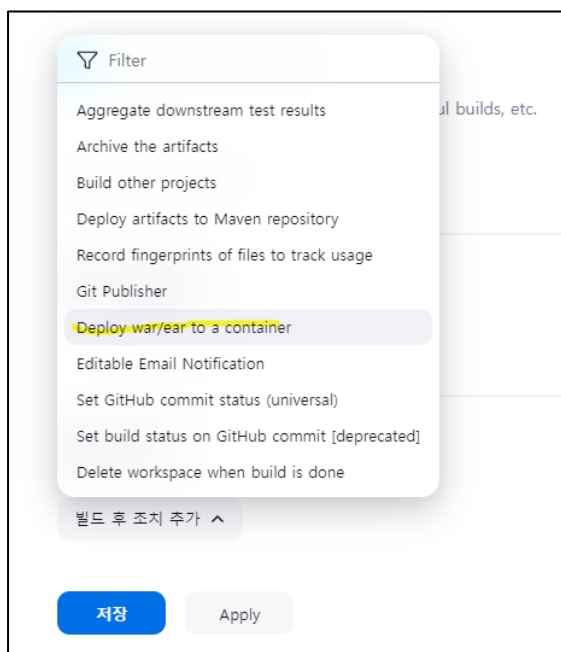
1) Tomcat Plugin 추가 (Jenkins 관리 > Plugins)

Available plugins 에서 Deploy 검색어 입력하고 Deploy to container 플러그인 설치하기



2) 빌드 후 조치 (현재 진행중인 Todo-BootApplication item 에서 설정)

Todo-BootApplication > 구성(Configuration) > General > 빌드 후 조치 > 빌드 후 조치 추가 선택 > Deploy war/ear to a container 선택



WAR/EAR files 항목에 ****/*.war** 입력하고 Context Path 항목은 비워둔다.

빌드 후 조치

Deploy war/ear to a container

WAR/EAR files ?

****/*.war** ****/*.war**

Context path ?

Containers

Add Container ▾

☐ Deploy on failure

Add Container 클릭 > Tomcat 9.x Remote 선택하기

Filter

- GlassFish 2.x
- GlassFish 3.x
- GlassFish 4.x
- JBoss AS 3.x
- JBoss AS 4.x
- JBoss AS 5.x
- JBoss AS 6.x
- JBoss AS 7.x
- Tomcat 4.x Remote
- Tomcat 5.x Remote
- Tomcat 6.x Remote
- Tomcat 7.x Remote
- Tomcat 8.x Remote
- Tomcat 9.x Remote**

Add Container ⬆

+ Add 클릭 > Jenkins 선택하기

빌드 후 조치

Deploy war/ear to a container

WAR/EAR files ?

**/*.war

Context path ?

Containers

Tomcat 9.x Remote

Credentials

- none -

+ Add

Jenkins

Tomcat 9 에 접근해서 배포하기 위한 권한을 설정해야 된다.

```
<role rolename="manager-gui"/>
<role rolename="manager-script"/>
<role rolename="manager-jmx"/>
<role rolename="manager-status"/>

<user username="admin" password="admin"
      roles="manager-gui,manager-script,manager-jmx,manager-status"/>
<user username="deployer" password="deployer" roles="manager-script"/>
<user username="tomcat" password="tomcat" roles="manager-gui"/>
```

Jenkins Credentials Provider: Jenkins

Domain

Global credentials (unrestricted)

Kind

Username with password

Scope ?

Global (Jenkins, nodes, items, all child items, etc)

Username ?

deployer

☐ Treat username as secret ?

Password ?

deployer

Credentials 항목에는 `deployer/*****` 선택하고 Tomcat URL 에는 <http://localhost:8090/> 입력한다.

최종 설정 화면은 다음과 같다.

빌드 후 조치

Deploy war/ear to a container

WAR/EAR files ?

**/*.war

Context path ?

Containers

Tomcat 9.x Remote

Credentials

deployer/*****

+Add

Tomcat URL ?

http://localhost:8090/

고급

저장 Apply

저장하고 지금 빌드 항목 선택.

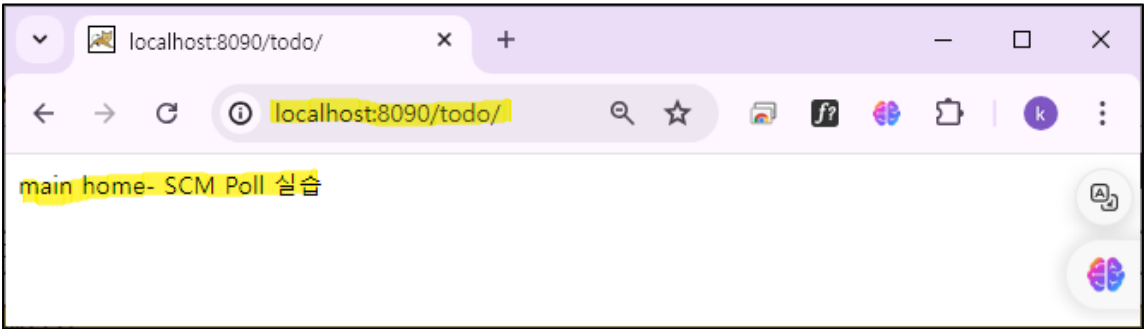
(빌드하기전에 반드시 Tomcat 이 실행되어 있어야 됨)

Build History		추인
Q	Filter builds...	/
✓ #2	2024. 1. 15. 오후 3:32	
✓ #1	2024. 1. 15. 오후 3:06	
Atom feed (전체) Atom feed (실패)		

빌드가 성공하면 다음과 같이 Tomcat 의 webapps에 todo.war 파일이 배포된다.

/host-manager	지정 안됨	Tomcat Host Manager Application	true	0	시작 중지 다시 로드 배치된 것을 제거 세션들을 만료시키기 idle 값 ≥ 30 분
/manager	지정 안됨	Tomcat Manager Application	true	1	시작 중지 다시 로드 배치된 것을 제거 세션들을 만료시키기 idle 값 ≥ 30 분
/todo	지정 안됨		true	0	시작 중지 다시 로드 배치된 것을 제거 세션들을 만료시키기 idle 값 ≥ 30 분

웹 브라우저에서 http://localhost:8090/todo/ 요청 화면은 다음과 같다.



6. Jenkins에 Poll SCM 설정하기

Github에 새롭게 commit 된 변경사항이 발생될 때 마다 자동으로 저장된 웹 어플리케이션을 참조해서 Jenkins에 war 저장하고 사용중인 Tomcat 서버에 배포까지 처리.

1) 빌드 유발 항목에서 Setup Poll SCM 활성화

(Todo-BootApplication > 구성 > 빌드 유발

빌드 유발

- ☒ Build whenever a SNAPSHOT dependency is built ?
- ☐ Schedule build when some upstream has no successful builds ?
- ☐ 빌드를 원격으로 유발 (예: 스크립트 사용) ?
- ☐ Build after other projects are built ?
- ☐ Build periodically ?
- ☐ GitHub hook trigger for GITScm polling ?
- ☒ Poll SCM ?
 - Schedule ?

`* * * * *`

⚠ Do you really mean "every minute" when you say "* * * * *"? Perhaps you meant "H * * * * *" to poll once per hour

Would last have run at 2024년 7월 26일 금요일 오후 4시 19분 20초 대한민국 표준시; would next run at 2024년 7월 26일 금요일 오후 4시 19분 20초 대한민국 표준시.
 - ☐ Ignore post-commit hooks ?

`*(분, 0~59) *(시, 0~23) *(일, 1~31) *(월, 1~12) *(요일, 0~6)`

Linux 의 cron job 스케줄러 샘플 예 >

@ 매일 새벽 4시에 /home/root/test.sh 명령어 실행하기

매일 새벽에 원하는 작업을 test.sh 이름의 쉘스크립트를 만들어 실행하는 방법입니다.

```
00 04 * * * /home/root/test.sh
```

@ 매주 화요일 오전 05시에만 특정 스크립트 실행하기

이번에는 매 주 한 번만 실행 할 경우의 방법입니다.

```
00 04 * * 2 /home/root/test.sh
```

@ 연속된 날짜 및 요일 실행하기

추가로 토요일부터 일요일까지 실행하려면? 이때는 **-기호를** 사용할 수 있습니다. 즉 토요일은 6이므로 6-7이 바로 토, 일 이 되죠.

```
00 00 * * 6-7 /home/root/test.sh
```

STS 에서 코드를 수정하고 Github에 push 한다.

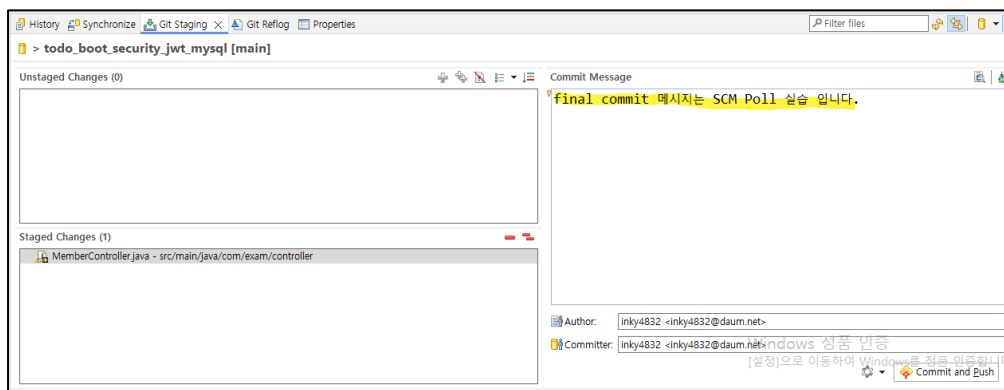
Push 하면 자동으로 Jenkins 가 변경사항을 체크해서 빌드까지 자동으로 해준다.

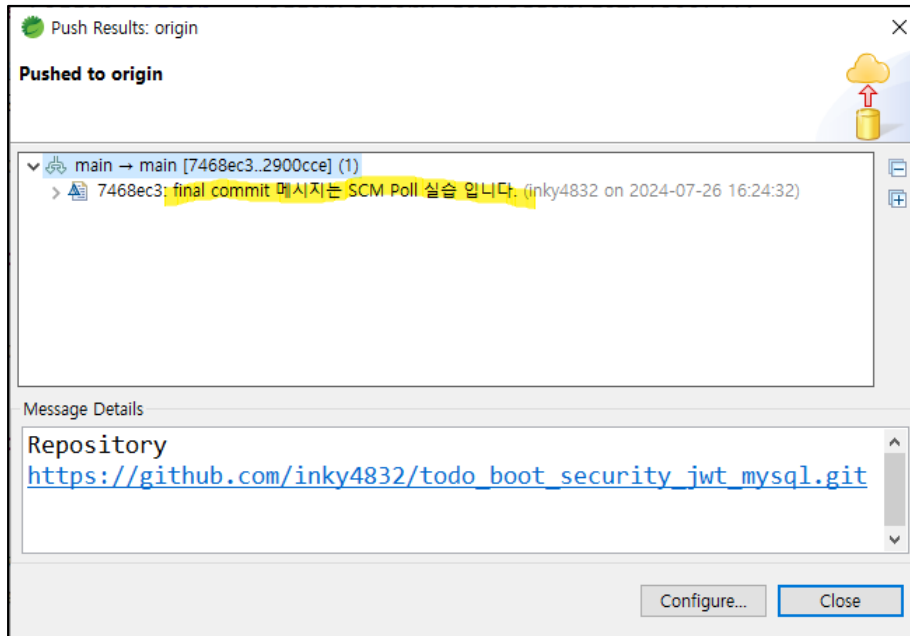
MemberController 수정하기

```
@GetMapping("/")
public String main( ) {

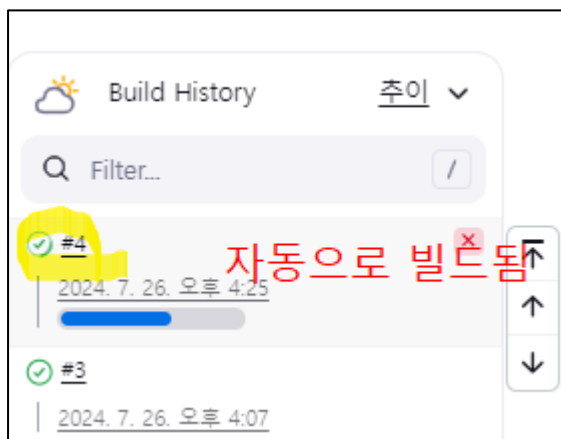
    return "main home- SCM Poll 실행. 자동으로 변경사항 인식해서 자동배포함";
}
```

Github에 push

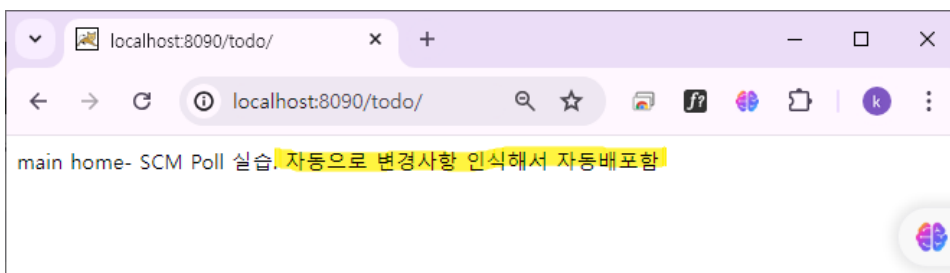




Jenkins 에서 자동으로 빌드됨.



다시 요청하기



7. ReactJS 프론트엔드 어플리케이션 배포

1) Ngnix 설치

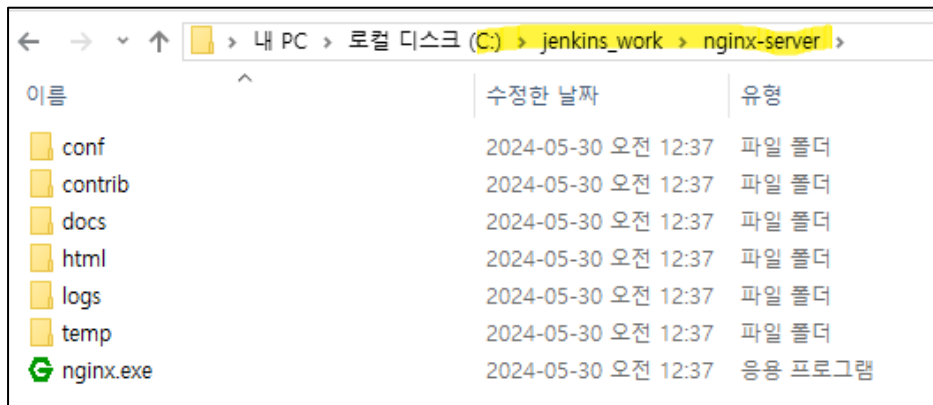
<https://nginx.org/en/download.html>

다음 화면에서 Stable version 의 Window 용을 다운로드 한다. (install 프로그램은 아니고 zip 형식)



2)zip 파일 압축 풀기

다음과 같은 디렉터리 구조로 nginx 압축파일을 풀자.



3)nginx 서버 설정 정보 확인

- conf/nginx.conf 파일 열기

```

server {
    listen 80;
    server_name localhost;

    #charset koi8-r;

    #access_log logs/host.access.log main;

    location / {
        root html;
        index index.html index.htm;
    }
}

```

nginx 기본포트

4) nginx 서버 시작

-nginx.exe 실행파일 더블 클릭.

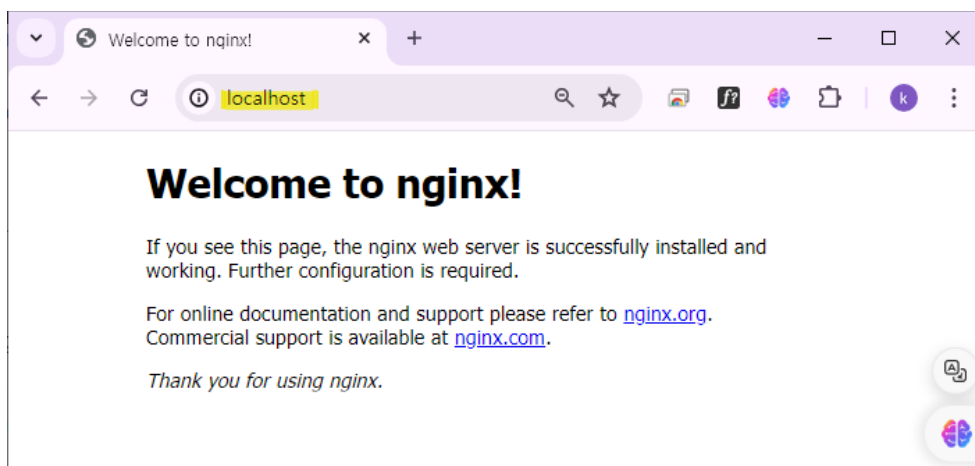
실행중인 확인은 작업 관리자에서 확인 가능.

이름	상태	CPU	메모리	디스크	네트워크	GPU	GPU 엔진	전력 사용량	전력
mysql.exe		9%	65%	0%	0%	2%			
nginx.exe(32비트)		0%	2.0MB	0MB/s	0Mbps	0%		매우 낮음	매우 낮음
nginx.exe(32비트)		0%	0.8MB	0MB/s	0Mbps	0%		매우 낮음	매우 낮음
nginx.exe(32비트)		0%	1.0MB	0MB/s	0Mbps	0%		매우 낮음	매우 낮음
NVIDIA Container		0%	5.5MB	0MB/s	0Mbps	0%		매우 낮음	매우 낮음

nginx 서버를 종료할 때는 작업관리자에서 프로세스를 명시적으로 종료 시킨다.

-웹 브라우저에서 요청

<http://localhost:80> 또는 <http://localhost> 사용.



5) nginx 서버에 Reactjs 어플리케이션을 명시적으로 배포하기

- VSC에서 reactjs 어플리케이션을 빌드하자.

npm run build 명령어 사용

```
C:\reactjs_study\my-app>npm run build

> my-app@0.1.0 build
> react-scripts build

Creating an optimized production build...
█
```

```
File sizes after gzip:

68.32 kB  build\static\js\main.138119b0.js
31.96 kB  build\static\css\main.a1674e4c.css
1.77 kB   build\static\js\453.ed3810f9.chunk.js

The project was built assuming it is hosted at /.
You can control this with the homepage field in your package.json.

The build folder is ready to be deployed.
You may serve it with a static server:

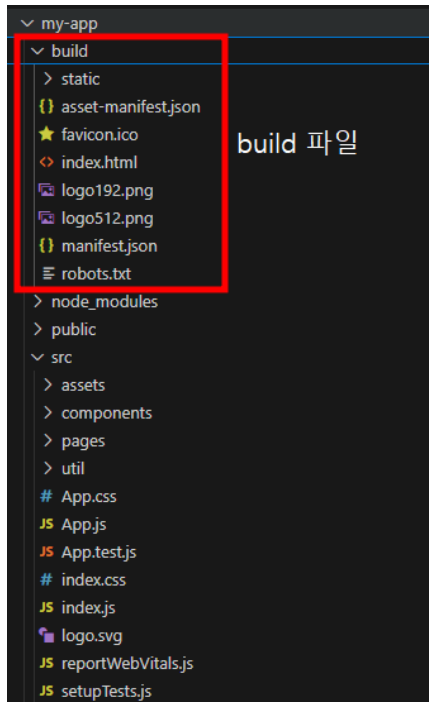
  npm install -g serve
  serve -s build

Find out more about deployment here:

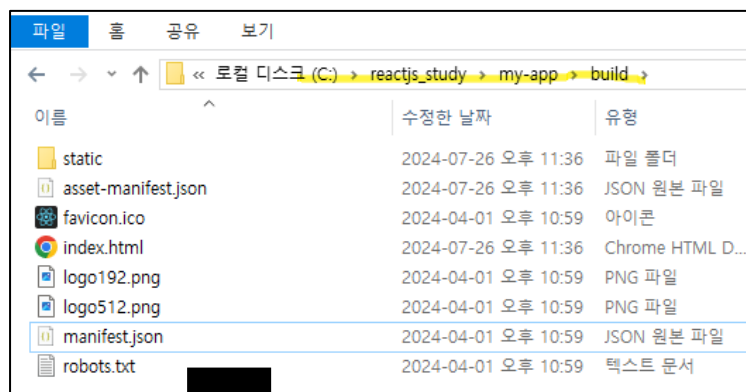
  https://cra.link/deployment

C:\reactjs_study\my-app>
```

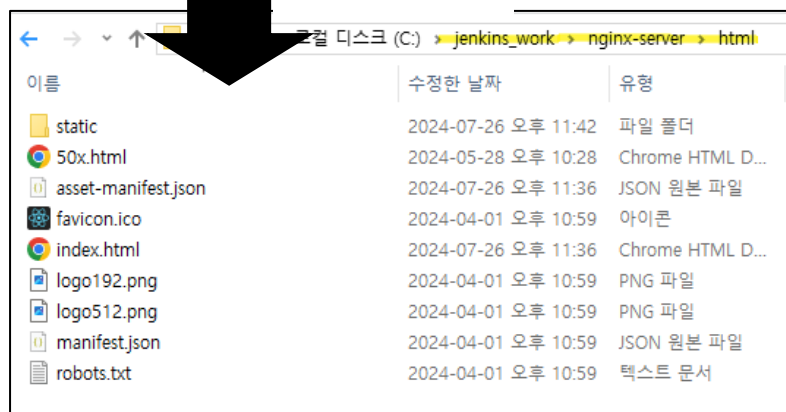
build 명령어로 생성된 파일들은 다음과 같다.



위의 build 폴더내의 파일들을 Jenkins에 배포하기 위해서 html 폴더에 복사한다.



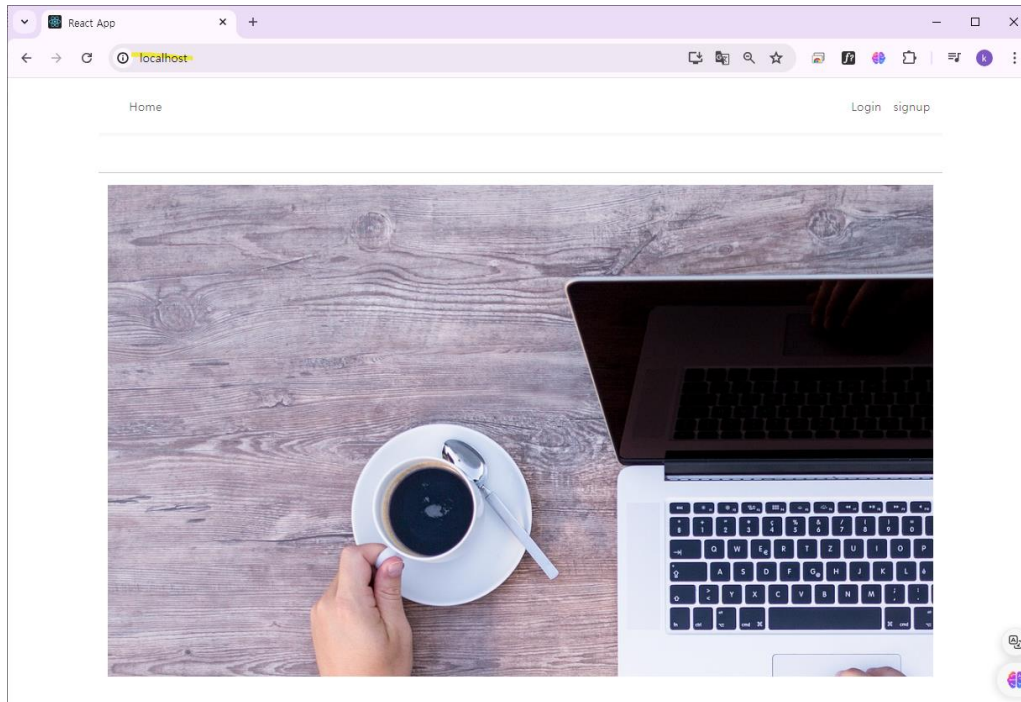
복사하기



-웹 브라우저에서 요청

<http://localhost:80> 또는 <http://localhost> 사용

실행결과는 Reactjs 로 만든 어플리케이션이 보임.



6) SpringBoot와 연동하기 위한 포워드 처리. (옵션 기능)

- nginx-server\conf\nginx.conf 파일 수정.

```
server {
    listen      80;
    server_name localhost;

    #charset koi8-r;

    #access_log logs/host.access.log main;

    location / {
        root    html;
        index   index.html index.htm;
    }

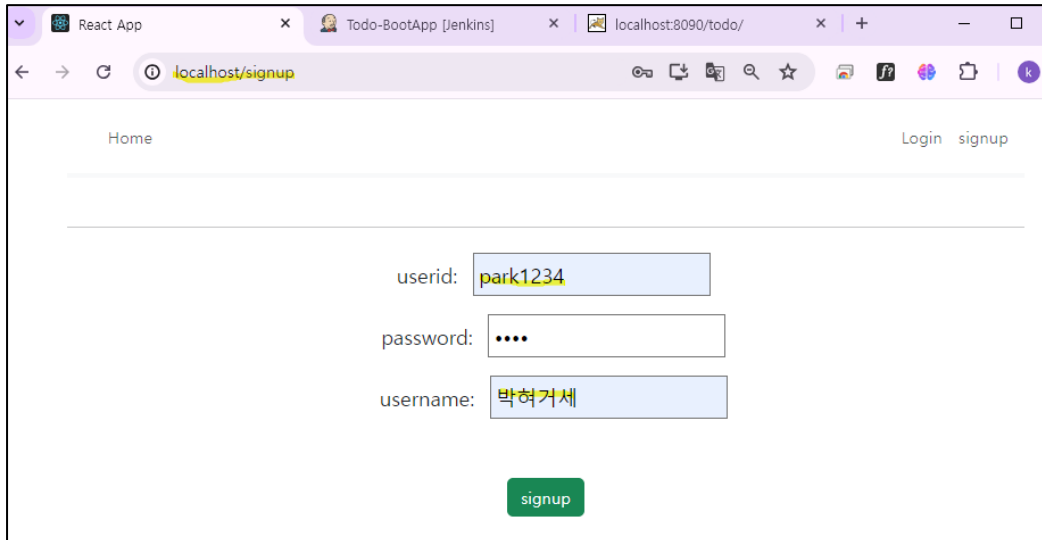
    # SpringBoot와 연동하기 위한 포워드 처리
    # Reactjs에서 /todo 경로로 요청하면 http://localhost:8090 으로 포워드 됨.
    location /todo {
        proxy_pass http://localhost:8090;
    }

    #error_page 404          /404.html;
```

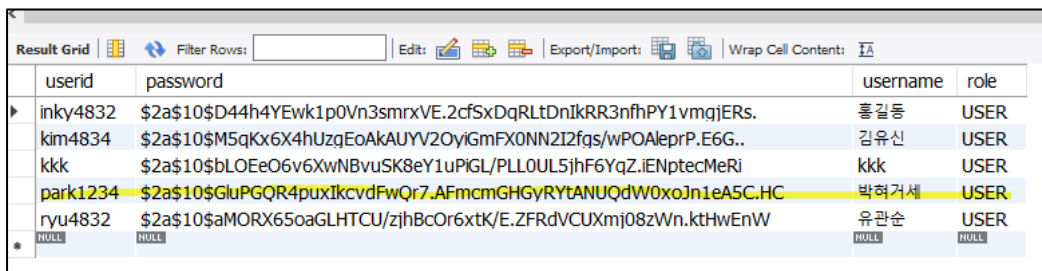
코드 추가하기

-nginx 서버 실행후 요청하기

회원 가입하기 위하여 <http://localhost/signup> 요청.



회원 가입 정보가 다음과 같이 MySQL 테이블에 저장됨.



	userid	password	username	role
▶	inky4832	\$2a\$10\$D44h4YEwk1p0Vn3smrxVE.2cfSxDqRLtDnIkRR3nfhPY1vmqjERs.	홍길동	USER
	kim4834	\$2a\$10\$M5qKx6X4hUzgEoAkAUyV2OyiGmFX0NN2I2fqs/wPOAleprP.E6G..	김유신	USER
	kkk	\$2a\$10\$bLOEeO6v6XwNBvuSK8eY1uPiGL/PLL0UL5jhF6YqZ.iENptecMeRi	kkk	USER
	park1234	\$2a\$10\$GluPGOR4pux1KcvdFwQr7.AFmcmGHGyRYtANUQdW0xoJn1eASC.HC	박혁거세	USER
	ryu4832	\$2a\$10\$aMORX65oaGLHTCU/zjhBcOr6xtK/E.ZFRdVCUXmj08zWn.ktHwEnW	유관순	USER
*				

7) Github (todo_react 원격 저장소)에 build 폴더 Push 하기

```
$ cd build
```

```
$ git init
```

```
$ git config --global core.autocrlf true
```

```
$ git add .
```

```
# commit 메시지는 반드시 "" (쌍따옴표) 사용할 것
```

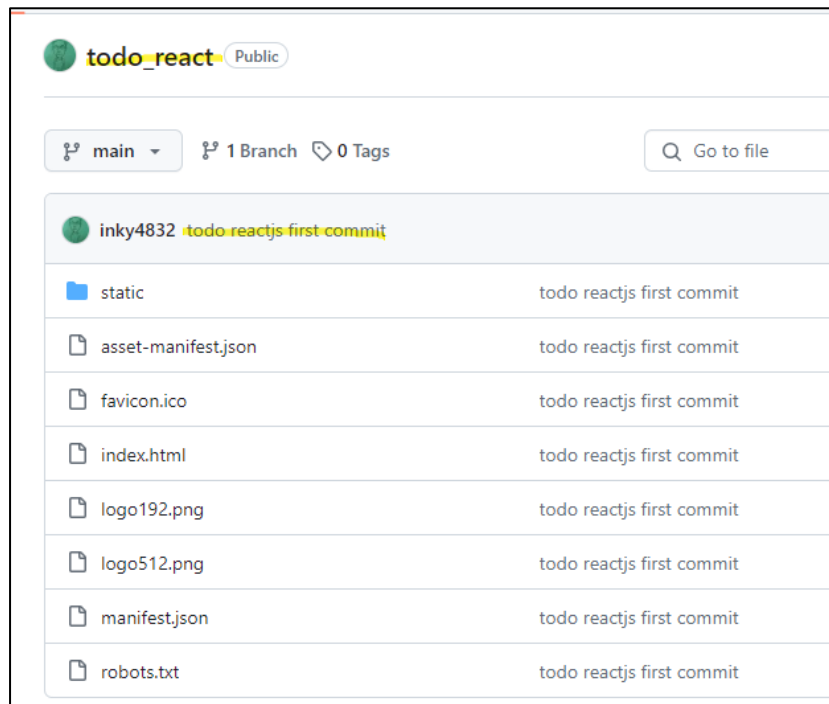
```
$ git commit -m "todo reactjs first commit"
```

```
$ git remote add origin https://github.com/inky4832/todo_react.git
```

```
$ git branch -M main
```

```
$ git push -u origin main    ( +main 지정하면 강제로 push )
```

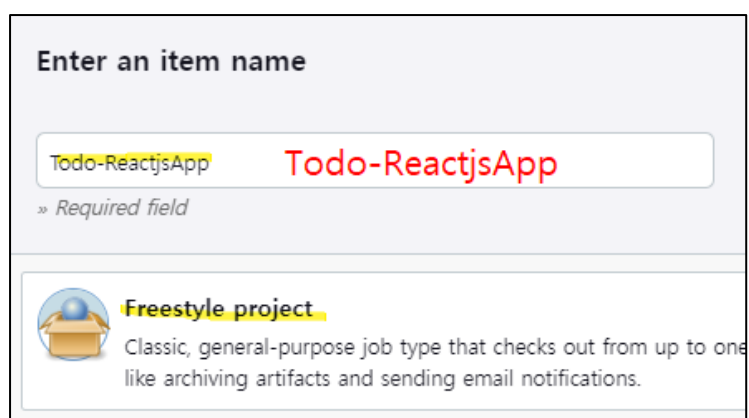
성공적으로 github에 push 하면 다음과 같이 todo_react 원격 저장소에 push된 파일들을 확인가능.



8) Jenkins 에서 Reactjs 어플리케이션 배포

가. 새로운 item 생성하기

Todo-ReactjsApp 이름의 Freestyle project 설정



나. 소스 코드 관리

Todo-ReactjsApp > 구성 > 소스 코드 관리

https://github.com/inky4832/todo_react.git

소스 코드 관리

☐ None

☒ Git ?

Repositories ?

Repository URL ?

https://github.com/inky4832/todo_react.git

Credentials ?

- none -

+ Add ▾

고급 ▾

Add Repository

Branches to build ?

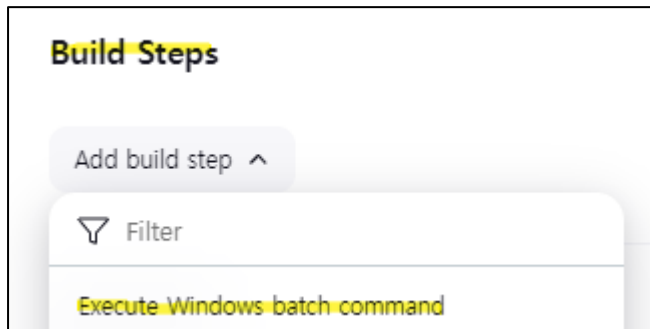
Branch Specifier (blank for 'any') ?

[*/main](#) [*/main](#)

Add Branch

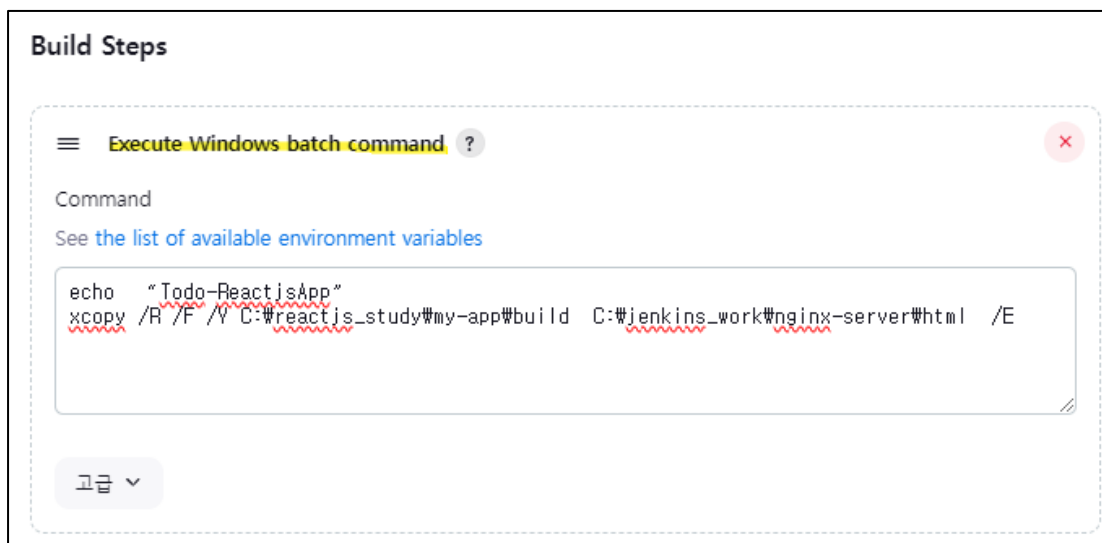
다. Build steps

Execute Windows batch command 선택



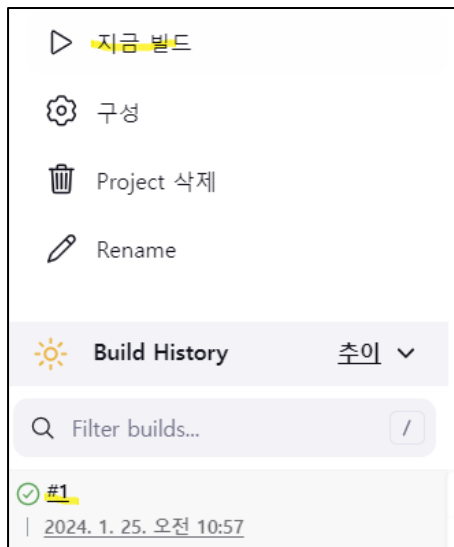
Command 항목에 다음 내용을 기입한다.

```
echo "Todo-ReactjsApp"  
xcopy /R /F /Y C:\reactjs_study\my-app\build C:\jenkins_work\nginx-server\html /E
```

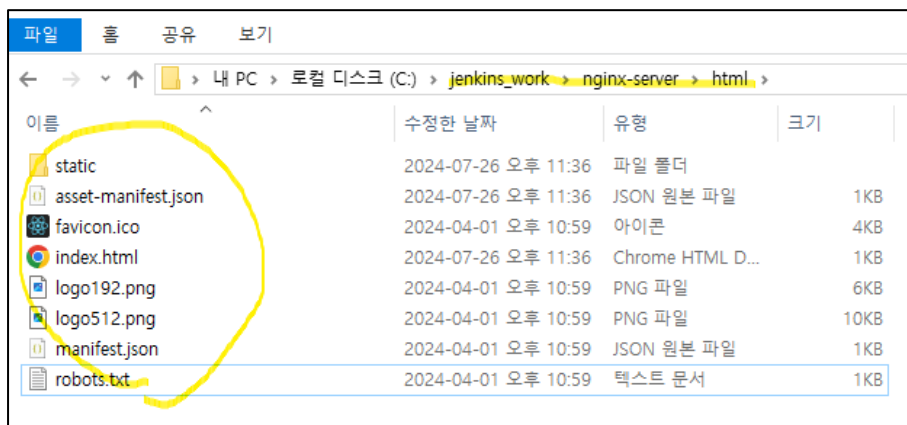


[저장] 버튼 클릭

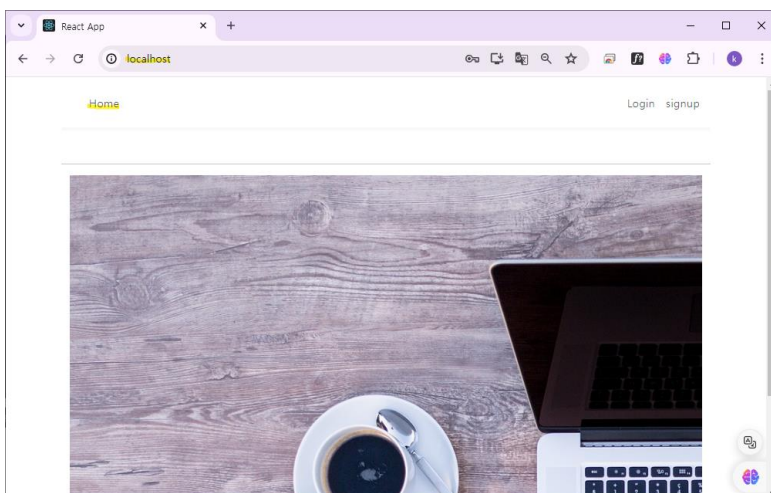
라. 지금 빌드 하기



마. 복사된 파일 확인하기

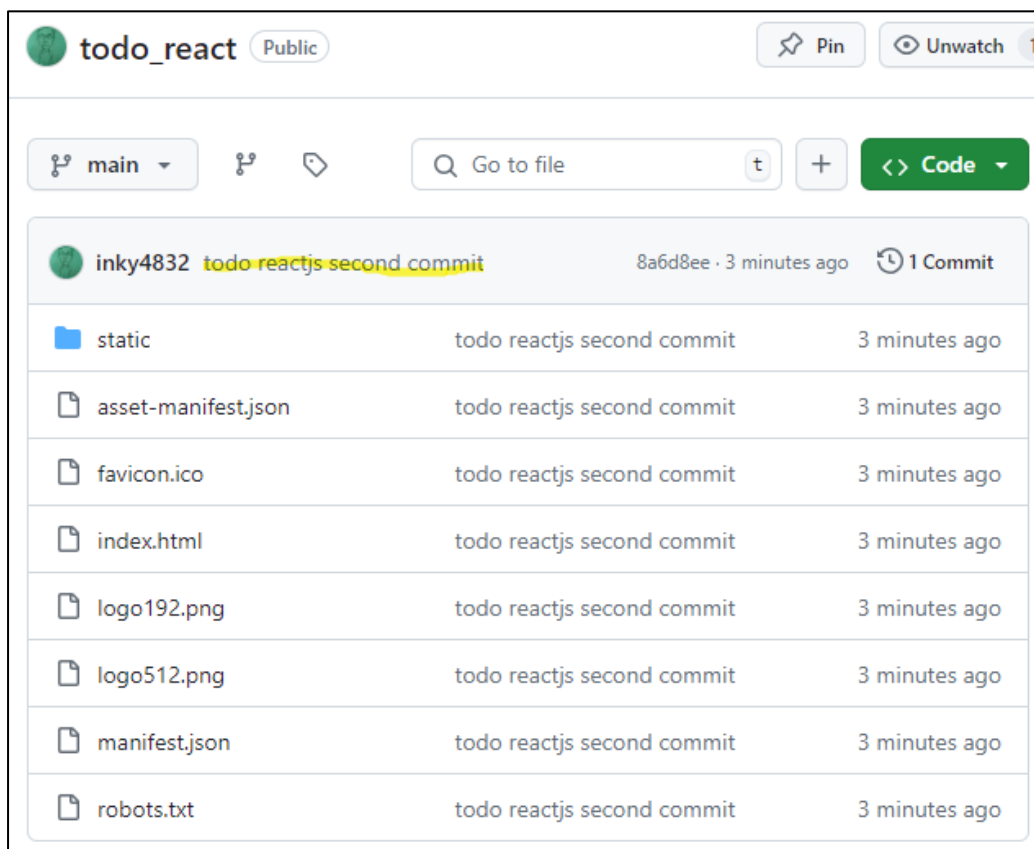


바. nginx 서버 요청하기



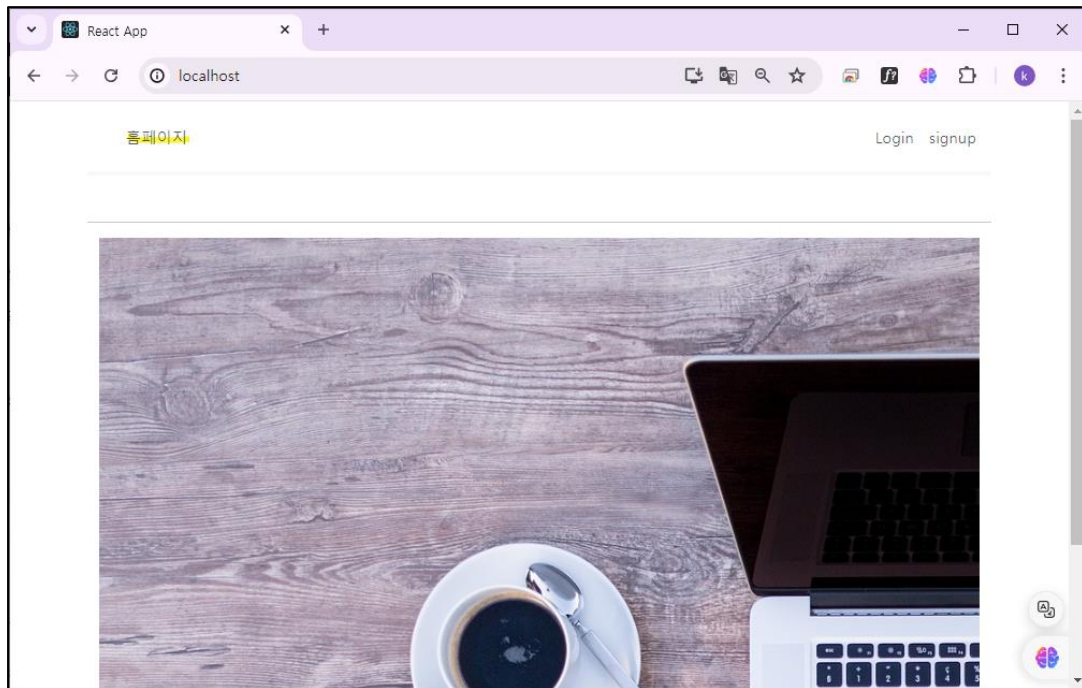
사. VSC 에서 코드 수정하고 변경사항 자동으로 배포하기

- 메인 페이지의 Home 링크를 홈페이지 링크로 변경.
- npm run build (**my-app** 디렉터리에서 실행)
- git init (**build** 디렉터리에서 실행)
- git config --global core.autocrlf true
- git add
- git commit -m "todo reactjs second commit"
- git remote add origin https://github.com/inky4832/todo_react.git
- git push -u origin +main (+main 지정하면 강제 push 됨)



아. 지금 빌드 하기 하고 nginx 서버 요청하기

다음과 같이 Home 링크가 홈페이지 링크로 수정되어 출력됨.



자. 빌드 유발 항목에서 Setup Poll SCM 활성화

(Todo-ReactjsApp > 구성 > 빌드 유발

빌드 유발

- ☒ Build whenever a SNAPSHOT dependency is built ?
- ☐ Schedule build when some upstream has no successful builds ?
- ☐ 빌드를 원격으로 유발 (예: 스크립트 사용) ?
- ☐ Build after other projects are built ?
- ☐ Build periodically ?
- ☐ GitHub hook trigger for GITScm polling ?
- ☒ **Poll SCM** ?

Schedule ?

*** * * * ***

⚠ Do you really mean "every minute" when you say "* * * * *"? Perhaps you meant "H * * * *" to poll once per hour
Would last have run at 2024년 7월 26일 금요일 오후 4시 19분 20초 대한민국 표준시; would next run at 2024년 7월 26일 금요일 오후 4시 19분 20초 대한민국 표준시.

☐ Ignore post-commit hooks ?

*** (분, 0~59) * (시, 0~23) * (일, 1~31) * (월, 1~12) * (요일, 0~6)**

차. VSC 에서 코드 수정하고 변경사항 자동으로 배포하기

- 메인 페이지의 홈페이지 링크를 HomePage 링크로 변경.
- npm run build (**my-app** 디렉터리에서 실행)
- git init (**build** 디렉터리에서 실행)
- git config --global core.autocrlf true
- git add
- git commit -m "todo reactjs third commit"
- git remote add origin https://github.com/inky4832/todo_react.git
- git push -u origin +main (+main 지정하면 강제 push 됨)

Jenkins 에서 자동으로 github의 변경 사항을 체크해서 자동으로 배포해줌.

github 화면

todo_react Public

Pin Unwatch 1

main

Go to file t + <> Code

inky4832	todo reactjs third commit	99eb09e · 2 minutes ago	1 Commit
static	todo reactjs third commit	2 minutes ago	
asset-manifest.json	todo reactjs third commit	2 minutes ago	
favicon.ico	todo reactjs third commit	2 minutes ago	
index.html	todo reactjs third commit	2 minutes ago	
logo192.png	todo reactjs third commit	2 minutes ago	
logo512.png	todo reactjs third commit	2 minutes ago	
manifest.json	todo reactjs third commit	2 minutes ago	
robots.txt	todo reactjs third commit	2 minutes ago	

nginx 서버 요청w

다음과 같이 HomePage 링크로 변경된 것을 확인 할 수 있음.

